Listing of Claims

This listing of claims replaces all prior versions, and listings, of claims in the application:

1. (Previously Presented) An execution unit for execution of multiple context threads, comprising:

an arithmetic logic unit to process data for executing threads;

control logic to control the operation of the arithmetic logic unit; and

a general purpose register set to store and obtain operands for the arithmetic logic unit, the register set comprising a plurality of two-ported random access memory devices assembled into banks, the register set comprising two effective read ports and one effective write port, wherein the effective write port comprises write ports of a pair of the two-ported random access memory devices, each bank being capable of performing a read and a write to two different words in the same processor cycle,

wherein the arithmetic logic unit can write to each bank in the general purpose register set using the one effective write port.

2. (Original) The execution unit of claim 1 wherein the register set is logically partitioned into a plurality of relatively addressable windows.

3. (Previously Presented) The execution unit of claim 2 wherein the number of windows of the register set is related to the number of threads that execute in the processor.

4. (Previously Presented) The execution unit of claim 1 wherein relative addressing allows an executing thread to access the register set relative to the starting point of a window.

5. (Previously Presented) The execution unit of claim 1 wherein the register set is absolutely addressable, where the register set may be accessed for an executing thread by providing an exact address.

6. (Previously Presented) The execution unit of claim 1 wherein the control logic further comprises:

context event switching logic fed by signals from a plurality of shared resources, the signals causing the context event switching logic to indicate that threads are either available or unavailable for execution.

7. (Previously Presented) The execution unit of claim 6 wherein the control logic addresses a first set of memory locations for storing a list of available threads that correspond to threads that are ready to be executed and a second set of memory locations for storing a list of unavailable threads that are not ready to be executed.

Claims 8-14. (Canceled)

15. (Previously Presented) A method for executing multiple context threads, comprising:

processing data for executing threads within an arithmetic logic unit;

operating control logic to control the arithmetic logic unit; and

storing and obtaining operands for the arithmetic logic unit within a general purpose register set comprising a plurality of banks of two-ported random access memory devices, the register set comprising two effective read ports and one effective write port, wherein the effective write port comprises write ports of a pair of the two-ported random access memory devices, the effective write port including a single write line to write to addresses in different banks of the plurality of banks, and each bank being capable of performing a read and a write to two different words in the same processor cycle.

16. (Previously Presented) The method of claim 15 wherein the register set is relatively addressable.

17. (Previously Presented) The method of claim 15 further comprising:

arranging the register set into a number of windows according to the number of threads that execute.

18. (Previously Presented) The method of claim 17 wherein storing and obtaining further comprises:

addressing the register set for an executing thread by providing a relative register address that is relative to the starting point of a window.

19. (Previously Presented) A processor unit comprising:

an execution unit for execution of multiple context threads, the execution unit comprising:

an arithmetic logic unit to process data for executing threads;

control logic to control the operation of the arithmetic logic unit;

a general purpose register set to store and obtain operands for the arithmetic logic unit, the register set comprising a plurality of two-ported random access memory devices, the register set comprising two effective read ports

and one effective write port, wherein the effective write port comprises write ports of a pair of the two-ported random access memory devices; and

a data link between the arithmetic logic unit and the one effective write port of the general purpose register set, wherein the data link allows the arithmetic logic unit to write to different two-ported random access memory devices in the general purpose register set through the one effective write port.

20. (Previously Presented) The processor of claim 19 wherein the register set is logically partitioned into a plurality of relatively addressable windows, where the number of windows of the register set is related to the number of threads that execute in the processor.

21. (Previously Presented) The processor of claim 20 wherein relative addressing allows an executing thread to access the register set relative to the starting point of a window.

22. (Previously Presented) The processor of claim 20 wherein the register set is absolutely addressable, where the register set may be accessed for an executing thread by providing an exact address.

23. (Previously Presented) The processor of claim 20 further comprising:

a set of memory locations for storing a list of available threads that are ready to be executed;

a set of memory locations for storing a list of unavailable threads that are not ready to be executed; and

context event switching logic fed by signals from a plurality of shared resources, the signals causing the context event switching logic to indicate which threads are either available or unavailable for execution.

24. (Previously Presented) The processor of claim 23 wherein execution of a context swap instruction causes a currently running thread to be swapped out to the list of unavailable threads and a thread from the list of available threads to begin execution within a single execution cycle.

25. (Previously Presented) The processor of claim 23 wherein execution of a context swap instruction specifies one of the signals and upon receipt of the specified signal causes a swapped out thread to be stored in the available thread memory set.

26. (Previously Presented) The processor of claim 23 wherein execution of a context swap instruction specifies a defer_one operation which causes execution of one more instruction and then causes a current context to be swapped out.

Claims 27-29. (Canceled)

30. (Previously Presented) The execution unit of claim 1 wherein the register set comprises a first number n of two-ported random access memory devices, a second number r of effective read ports, and a third number w of effective write ports, where $n \geq 2$, $2 \leq r \leq n$, and $2 \leq w \leq n-1$.

31. (Previously Presented) The method of claim 15 wherein storing and obtaining operands comprises storing and obtaining operands within the general purpose register comprising a first number n of two-ported random access memory devices, a second number r of effective read ports, and a third number w of effective write ports, where $n \geq 2$, $2 \leq r \leq n$, and $2 \leq w \leq n-1$.

32. (Previously Presented) The execution unit of claim 19 wherein the general purpose register set comprises a first number n of two-ported random access memory devices, a second number r of effective read ports, and a third number w of effective write ports, where $n \geq 2$, $2 \leq r \leq n$, and $2 \leq w \leq n-1$.

33. (Previously Presented) The execution unit of claim 1 wherein memory addresses of the pair of the two-ported random access memory devices are interleaved.

34. (Previously Presented) The processor of claim 19 wherein memory addresses of the pair of the two-ported random access memory devices are interleaved.

35. (Previously Presented) The method of claim 15 wherein storing the operands for the arithmetic logic unit comprises writing to interleaved memory addresses of the pair of the two-ported random access memory devices over the single write line.